Thoughts of Algorithms

【算法27】硬币面值组合问题

问题描述

假设我们有8种不同面值的硬币｛1，2，5，10，20，50，100，200｝，用这些硬币组合够成一个给定的数值n。例如n=200，那么一种可能的组合方式为 200 = 3 * 1 + 1*2 + 1*5 + 2*20 + 1 * 50 + 1 * 100. 问总过有多少种可能的组合方式？（这道题目来自著名编程网站ProjectEuler，点击 这里 查看原题目）类似的题目还有：

[华为面试题] 1分2分5分的硬币三种，组合成1角，共有多少种组合

[创新工厂笔试题] 有1分，2分，5分，10分四种硬币，每种硬币数量无限，给定n分钱，有多少中组合可以组成n分钱

问题分析

给定一个数值sum，假设我们有m种不同类型的硬币{V1, V2, ..., Vm}，如果要组合成sum，那么我们有

$$sum = x1 * V1 + x2 * V2 + ... + xm * Vm$$

求所有可能的组合数，就是求满足前面等值的系数{x1, x2, ..., xm}的所有可能个数。

【思路1】 当然我们可以采用暴力枚举，各个系数可能的取值无非是x1 = {0, 1, ..., sum / V1}, x2 = {0, 1, ..., sum/ V2}等等。这对于硬币种类数较小的题目还是可以应付的，比如华为和创新工厂的题目，但是复杂度也很高O（sum/V1 * sum/V2 * sum/V3 * ...）

【思路2】 从上面的分析中我们也可以这么考虑，我们希望用m种硬币构成sum，根据最后一个硬币Vm的系数的取值为无非有这么几种情况，xm分别取｛0, 1, 2, ..., sum/Vm｝，换句话说，上面分析中的等式和下面的几个等式的联合是等价的。

$$sum = x1 * V1 + x2 * V2 + ... + 0 * Vm$$

$$sum = x1 * V1 + x2 * V2 + ... + 1 * Vm$$

$$sum = x1 * V1 + x2 * V2 + ... + 2 * Vm$$

$$...$$

$$sum = x1 * V1 + x2 * V2 + ... + K * Vm$$

其中K是该xm能取的最大数值K = sum / Vm。可是这又有什么用呢？不要急，我们先进行如下变量的定义：

dp[i][sum] = 用前i种硬币构成sum 的所有组合数。

那么题目的问题实际上就是求dp[m][sum]，即用前m种硬币（所有硬币）构成sum的所有组合数。在上面的联合等式中：当xn=0时，有多少种组合呢？ 实际上就是前i-1种硬币组合sum，有dp[i-1][sum]种！ xn = 1 时呢，有多少种组合？ 实际上是用前i-1种硬币组合成(sum - Vm)的组合数，有dp[i-1][sum -Vm]种; xn =2呢， dp[i-1][sum - 2 * Vm]种，等等。所有的这些情况加起来就是我们的dp[i][sum]。所以：

$$dp[i][sum] = dp[i-1][sum - 0*Vm] + dp[i-1][sum - 1*Vm]$$

+ dp[i-1][sum - 2*Vm] + ... + dp[i-1][sum - K*Vm]; 其中K = sum / Vm

换一种更抽象的数学描述就是：

$$dp[i][sum] = \sum_{k=0}^{sum/Vm} dp[i-1][sum - k * Vm]$$

通过此公式，我们可以看到问题被一步步缩小，那么初始情况是什么呢？如果sum=0，那么无论有前多少种来组合0，只有一种可能，就是各个系数都等于0；

dp[i][0] = 1 // i = 0, 1, 2, ... , m

如果我们用二位数组表示dp[i][sum]，我们发现第i行的值全部依赖与i-1行的值，所以我们可以逐行求解该数组。如果前0种硬币要组成sum，我们规定为dp[0][sum] = 0.

程序源码

通过上面的讨论，我们最终可以写出下面的代码来求解该类问题：

```
/*
 * Filename :coins.cpp
 * Description: solve coin combinations using dynamic programing
 * Complier: g++
 * Author: python27
 */
#include <iostream>
#include <string>
#include <cmath>
#include <vector>

using namespace std;

/*************************************************************
 * coin Combinations: using dynamic programming
 *
 * Basic idea:
 * dp[i][j] = sum(dp[i-1][j-k*coins[i-1]]) for k = 1,2,..., j/coins[i-1]
 * dp[0][j] = 1 for j = 0, 1, 2, ..., sum
 *
 * Input:
 * coins[] - array store all values of the coins
 * coinKinds - how many kinds of coins there are
 * sum - the number you want to construct using coins
 *
 * Output:
 * the number of combinations using coins construct sum
 *
 * Usage:
 * c[3] = {1, 2, 5};
 * int result = coinCombinations(c, 3, 10);
 *
 *************************************************************/
int coinCombinations(int coins[], int coinKinds, int sum)
{
    // 2-D array using vector: is equal to: dp[coinKinds+1][sum+1] = {0};
    vector<vector<int> > dp(coinKinds + 1);
    for (int i = 0; i <= coinKinds; ++i)
    {
        dp[i].resize(sum + 1);
    }
    for (int i = 0; i <= coinKinds; ++i)
    {
        for (int j = 0; j <= sum; ++j)
        {
            dp[i][j] = 0;
        }
    }

    //init: dp[i][0] = 1; i = 0, 1, 2 ..., coinKinds
    //Notice: dp[0][0] must be 1, althongh it make no sense that
    //using 0 kinds of coins construct 0 has one way. but it the foundation
    //of iteration. without it everything based on it goes wrong
    for (int i = 0; i <= coinKinds; ++i)
    {
```

```
56          dp[i][0] = 1;
57      }
58
59      // iteration: dp[i][j] = sum(dp[i-1][j - k*coins[i-1]])
60      // k = 0, 1, 2, ... , j / coins[i-1]
61      for (int i = 1; i <= coinKinds; ++i)
62      {
63          for (int j = 1; j <= sum; ++j)
64          {
65              dp[i][j] = 0;
66              for (int k = 0; k <= j / coins[i-1]; ++k)
67              {
68                  dp[i][j] += dp[i-1][j - k * coins[i-1]];
69              }
70          }
71      }
72
73      return dp[coinKinds][sum];
74  }
75
76  int main()
77  {
78      int coins[8] = {1, 2, 5, 10, 20, 50, 100, 200};
79      int sum = 200;
80      int result = coinCombinations(coins, 8, 200);
81      cout << "using 8 kinds of coins construct 200, combinations are: " << endl;
82      cout << result << endl;
83      return 0;
84  }
```

聪明的读者或许已经发现，在算法的描述中说明用动态规划的方法来求解此问题，什么？动态规划，我们什么时候用动态规划了？哈哈，在我们写出递归公式并且给出初始解的时候，我们就已经在用动态规划了。

动态规划的基本思想就是将待求解问题分解为若干子问题，（如本题中我们将dp[i][j]分解为若干dp[i-1][j-x]的问题），先求解这些子问题并将结果保存起来(我们用dp[][]二维数组保存子结果)，若在求解较大的问题时用到较小子问题的结果，可以直接取用(求dp[i][j]时用dp[i-1][x]的结果)，从而免去重复计算。动态规划是一种非常强大的算法思想，无论做过多少动态规划的题目，下一次依然会被动态规划的强大所震撼。随后的博客中，我们会更多的接触动态规划。你可以在后面的参考文献中找到更多有用的资源。

参考文献

[1] ProjectEuler: http://projecteuler.net/problem=31

[2] Topcoder Algorithm tutorial: http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=dynProg

[3] Sanjoy Dasgupta. 算法概论. 清华大学出版社，2008: 173 - 193.

[4] Thomas H. Cormen, et al. 算法导论. 机械工业出版社，2011: 192 - 212.

分类：每天一算法

标签：算法，动态规划

« 上一篇： 【Git】Git学习手册
» 下一篇： 【C++】C++中的操作符重载

posted @ 2013-09-05 17:51 python27 阅读(6279) 评论(4) 编辑 收藏

评论列表

#1楼 2013-11-05 10:47 Jack47
写的挺不错的，顶！
支持(0) 反对(0)

#2楼 2013-11-05 15:25 Jack47
注释中，第19行有错吧?
* dp[0][j] = 1 for j = 0, 1, 2, ..., sum
应该是
* dp[0][j] = 0 for j = 1, 2, ..., sum
* dp[i][0] = 1 for i = 0, 1, 2, ..., sum
支持(0) 反对(0)

#3楼 2013-11-17 18:53 六月心悸
代码用c++平台运行有错误
d:\c c++编程平台\showcoins.cpp(42) : error C2374: 'i' : redefinition; multiple initialization
d:\c c++编程平台\showcoins.cpp(38) : see declaration of 'i'
d:\c c++编程平台\showcoins.cpp(54) : error C2374: 'i' : redefinition; multiple initialization
d:\c c++编程平台\showcoins.cpp(38) : see declaration of 'i'
d:\c c++编程平台\showcoins.cpp(75) : error C2601: 'main' : local function definitions are illegal
d:\c c++编程平台\showcoins.cpp(83) : fatal error C1004: unexpected end of file found
执行 cl.exe 时出错.

showcoins.obj - 1 error(s), 0 warning(s)
支持(0) 反对(0)

#4楼 [楼主] 2014-04-10 15:46 python27
@ 六月心悸
你使用的应该是VC6，该编译器将for(int i = 0; i <n; ++i)中的变量解释为for循环之外的变量，所以显示redefinition，请尝试用VS2010或Gcc编译器。谢谢支持！
支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

最新IT新闻:
· 京东商城运费标准调整 提高免运费门槛 加收重量费
· 谷歌想在海上放飞风筝，用海风给数据中心供能散热
· 亚洲首次！日本命名113号元素为Nihonium
· 网曝京东高管禁止员工用微信和供应商沟通
· 乐视冯幸专访：明年我们拒绝补贴价格做手机
» 更多新闻...

最新知识库文章：

· 高质量的工程代码为什么难写
· 循序渐进地代码重构
· 技术的正宗与野路子
· 陈皓：什么是工程师文化?
· 没那么难，谈CSS的设计模式
» 更多知识库文章...